|  |  |
| --- | --- |
| Ponctuel 14 | **Les listes chaînées  (du point de vue logique)** |

Pour les exercices qui suivent, vous n’avez PAS accès aux détails d’implantation d’une liste (donc pas d’accès aux champs *info* et *suivant*). Par contre, vous disposez des opérations suivantes présentées en classe :

|  |  |
| --- | --- |
| *liste.* ***initialiser****( )* | *donnee 🡨 liste.****supprimerParIndice****( indice )* |
| *liste****.ajouterDebut****(donnee)* | *entier 🡨 liste.****longueur****( )* |
| *liste****.ajouterFin****(donnee)* | *chaine 🡨 liste.****enChaine****()* |
| *booléen 🡨 liste.****ajouter****(indice, donnee)* | *booléen 🡨 liste.****contient****(donnee)* |
| *donnee 🡨 liste.****supprimerDebut****( )* | *booléen 🡨 liste.****supprimerParValeur****( donnee )* |
| *donnee 🡨 liste.****supprimerFin****( )* |  |

1. Dites ce qui sera affiché après l’exécution de l'algorithme suivant, appliqué sur une liste de nombres entiers:

liste.initialiser( )

pour k🡨 1 à 10

liste.ajouterDebut( k )

fin pour

liste. supprimerParValeur ( 3 )

x 🡨 liste. supprimerParIndice ( 3 )

afficher "x=" , x

liste.ajouter(1, 99)

afficher liste.enChaine( )

1. On remarque que *supprimerParIndice* retourne une donnée, tandis que *supprimerParValeur* retourne un booléen. Selon-vous pourquoi en est-il ainsi?

1. A l'aide des opérations du tableau ci-haut, écrivez un algorithme pour ajouter à une liste existante tous les nombres entiers de 1 à 10 qui ne s'y trouvent pas déjà.